Title: Computer vision guidance for precise movement in commercial drones

Name: Alexander Ryan Pollard

Computer Vision Guidance for Precise Movement in Commercial Drones.

Volume One

Alexander Ryan Pollard

A Thesis Submitted to the University of Bedfordshire, in Fulfilment of the Requirements for the Degree of Master of Sciences by Research

University of Bedfordshire

Institute of Research in Applicable Computing (IRAC)

October 2017

Author's Declaration

I, Alexander Ryan Pollard, declare that this thesis and the work presented in it are my own and it has been generated by me as the result of my own original research.

I confirm that:

1. This work has been done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously has been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have cited the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done before myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Either none of this work has been published before submission, or parts of this work have been published as indicated on page 2.

Abstract

This project aims to pave the way for completely autonomous drone flight, starting with using computer vision to guide a drone. The project will involve using computer vision to direct a drone around a room to find its goal location. The current aims and objectives are:

- To implement computer control for drones and use this to run the program through the drone;
- To get the drone to recognize directions given by arrows using computer vision and to translate arrows seen into lateral directions;
- To accurately land the drone safely on or as close to a final goal position as possible.

Acknowledgements

I would like to use this section to thank friends, family, colleagues and university staff for their support during the four years of study at university that have led to this thesis.

Firstly, I would like to thank my supervisors, Dayou Li and Renxi Qui for their supervision during my research. Without their advice the project would have been far larger and as a result would have never been finished on time.

Secondly, I would like to thank my mother and father, Lisa Siddall and David Pollard, for their emotional and financial support that has allowed me to focus on my studies.

Finally, I would like to thank the following for also being there for me during my studies, helping me unwind and for all the joy they have brought that ultimately made the time at university enjoyable:

Stephen Chase, Khaqan Rana, Hannah Slack, Nicole Wild, Ivo Silva, Sheringham ‘Chip’ Reynolds, Harry Sambrook, Lauren Soley, Luke Hemmings, Callum Kauntzie-Cockburn, Alexandra McCarthy, Joe Newitt, Daniela Lopes.

List of Figures:

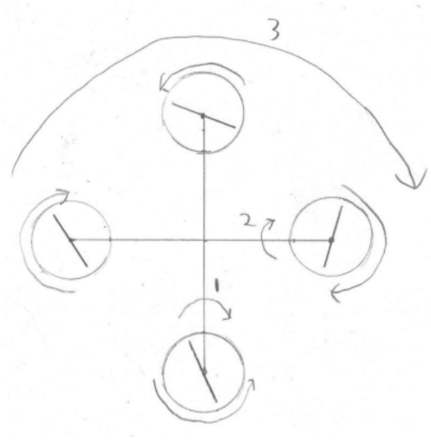


Figure 1, Alex Pollard (2017) Author.

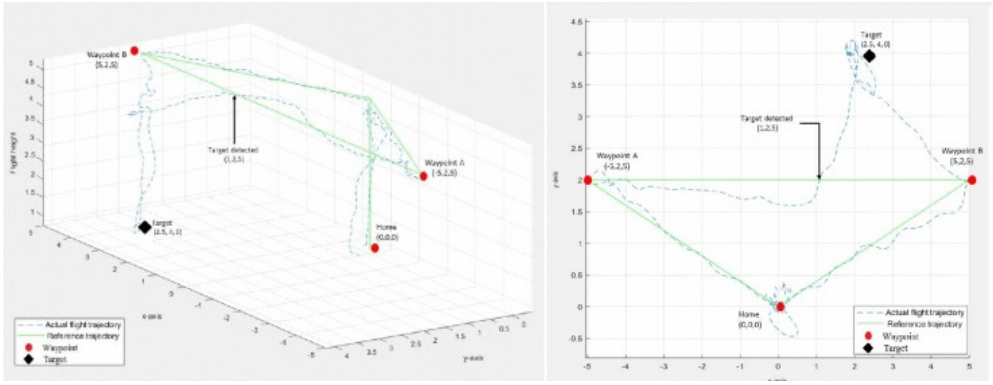


Figure 2, Alsalam, Morton, Campbell, Gonzalez (2017)

Figure 3, Alex Pollard (2017) Author

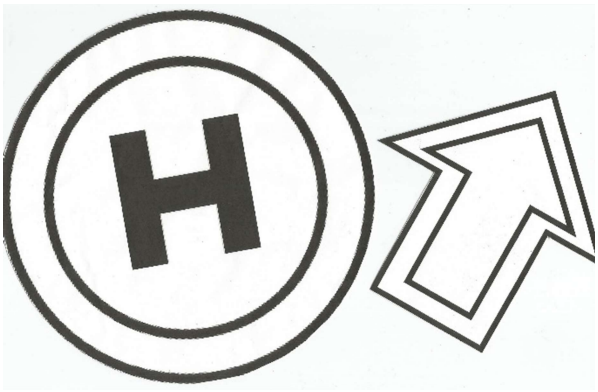


Figure 4, Alex Pollard (2017) Author

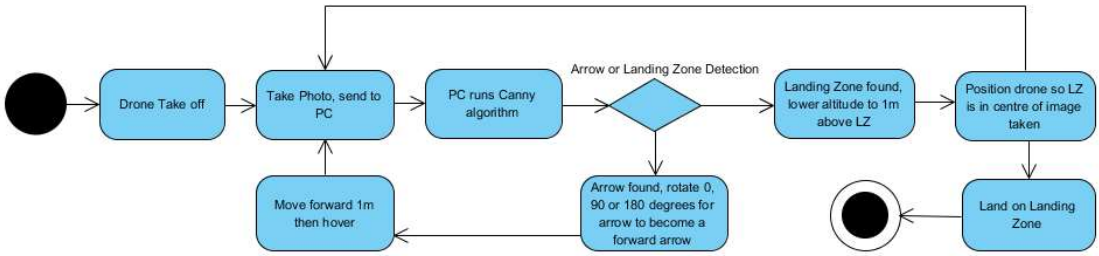
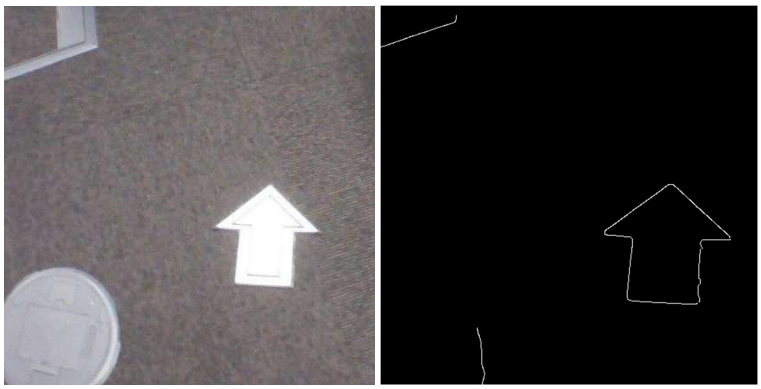


Figure 5, Alex Pollard (2017)



Figures 6 and 7, Alex Pollard (2017) Author

List of Tables:

Attempt number	Time (s)
1	1.50
2	1.30
3	1.44
4	1.48
5	1.46
6	1.47
7	1.44
8	1.49
9	1.45
10	1.44

Table 1, Alex Pollard (2017) Author

List of Formulas:

$FD_t(x,y) = |\, I_t(x,y) - I_{t-1}(x,y)$ (i), Kamate and Yilmazer (2015).

$FG_t(x,y) = \begin{cases} 1 \;\; if \;\; FD_t(x,y) > T1 \\ 0 \qquad\qquad else \end{cases}$ (ii), Kamate and Yilmazer (2015).

$BM_t(x,y) = \begin{cases} I_t(x,y) \; if \; CF_r > TF_r \\ 0 \qquad\qquad else \end{cases}$ (iii), Kamate and Yilmazer (2015).

$D_1(x,y) = |I_1(x,y) - BM_{t-1}(x,y)|$ (iv), Kamate and Yilmazer (2015).

$FG_t(x.y) = \begin{cases} 1 \; if \; D_t(x,y) \geq Th_{Ad}, t. \\ 0 \qquad\qquad else \end{cases}$ (v), Kamate and Yilmazer (2015).

$C_n^b = C_2^b C_1^2 C_n^1$

$$= \begin{bmatrix} \cos B & 0 & sinB \\ 0 & 1 & 0 \\ \sin B & 0 & \cos B \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cosY & sinY \\ 0 & -sinY & cosY \end{bmatrix} \begin{bmatrix} \cos a & -\sin a & 0 \\ \sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

=

$$\begin{bmatrix} \cos B \cos a + \sin B \sin a \sin y & -\cos B \sin a + \sin B \cos a \sin y & -\sin B \cos y \\ \sin a \cos y & \cos a \cos y & \sin y \\ \sin B \cos a - \cos B \sin a \sin y & -\sin B \; sin \; a - \cos B \cos a \sin y & \cos B \cos y \end{bmatrix}$$

(vi), Wei (2016)

C =

$$\begin{bmatrix} (a^2 + b^2 - c^2 - d^2) & 2(bc - ad) & 2(bd + ac) \\ 2(bc + ad) & (a^2 - b^2 + c^2 - d^2 & 2(cd - ab) \\ 2(bd - ac) & 2(cd + ad) & (a^2 - b^2 - c^2 + d^2) \end{bmatrix}$$

(vii), Wei (2016)

$$V = U + AT$$

$$S = UT + \frac{1}{2}AT^2$$

$$S = \frac{1}{2}(U + V)T$$

$$V^2 = U^2 + 2AS$$

$$S = VT - \frac{1}{2}AT^2$$ (viii), Equations of motion (SUVAT)

$$H_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right); 1 \leq i, j \leq 2k+1$$ (ix), John F. Canny (1986)

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$ (x), John F. Canny (1986)

List of Abbreviations

FPS	-	Frames Per Second
LXJS	-	Lisbon JavaScript Conference
UKAB	-	United Kingdom Airprox Board
PPE	-	Personal Protective Equipment
UAV	-	Unmanned Aerial Vehicle
OODA	-	Observation, Orientation, Decision, Action
HTML	-	Hypertext Mark-up Language

Chapter One: Introduction

1.1 Motivation

Drone technology is being proposed for many uses now. Amazon and UPS have begun testing to use drones to deliver parcels to customers, musicians such as Muse and Lady Gaga used drones during their performances in 2016 to create light shows and shapes in the sky, and drone racing is becoming an increasingly popular sport. Every year, more drones are becoming available for public use, and not every member of the public is fully aware of the laws and regulations surrounding them. According to The Guardian newspaper, UK police reported a spike in drone related incidents in 2015, with eighty reported incidents going from twenty-one incidents in 2014 (Quinn 2015). In addition to the information that drone related incidents are on the rise, the number of drones being flown near to airports causing risks to planes are also on the rise. In February 2017, it was reported by the UKAB (United Kingdom Airprox Board) of three separate occasions where drones were flown within one hundred and fifty feet of aircraft taking off from Heathrow airport, London.

With this information in mind, drone safety must improve and the technology behind drones needs to advance to match its growing interest. Preventative measures for keeping drones away from airports and buildings are needed. While this task is important, the solution propositioned in this paper will be a final line of defence for drones to avoid contact with buildings, other aircraft and people. The Propositioned solution is to incorporate computer vision into drones and to fly the drones autonomously using the computer vision as a guide. This idea can be incorporated into drones and be a failsafe if a drone sees it is too close to something, allowing it to take control of itself and move itself to a safer location, away from the airspace around an airport, building or tree for example.

The author accepts that other solutions such as keeping drones away from such areas in the first place would work far better, however there would be incidents where a drone still enters this airspace. When it comes to health and safety, there are 5 categories in risk control: Elimination, Substitution, Engineering Controls, Signage and Personal Protective Equipment. Elimination would not, at this stage, be completely achievable as it would be difficult to completely remove drones from the world. It has at least illegal to fly drones in such areas. Substitution would require replacing all drones with a safer model, or making all drones conform to one standard, which could prove to be very costly and ramp up the price of all drones. The final stage relevant would be the engineering controls, by making the control for drones safer. This is the final line reasonable to the use of drones as drones aren't allowed to be flown within fifty feet of other members of the public or within the designated no fly zones,

generally around heavily populated areas or airports. PPE would also be unrealistic as it would require the entire population to wear PPE at all times.

Although there is talk of introducing licensing for drone use, similar to that of a driving test, this is not in effect yet. Therefore, to protect people from those flying drones from outside the confines of the law the technology inside all drones must improve to protect the public from this minority of users.

1.2 Aims and Objectives

To achieve this, a reliable form of computer vision must be formed that can respond accordingly and that can fly itself away from these threats. This project will use computer vision to move a drone autonomously around a room to find a goal location. In a final iteration of this research a drone would recognise threats such as an incoming plane and calculate the ideal path to put as much distance from itself and the oncoming plane. However, in this first volume the computer vision will be used to recognise simple shapes to lay the ground work for further research. The aims and objectives for this project are therefore as follows:

- To implement computer control for drones and use this to run the program through the drone;
- To get the drone to recognize directions given by arrows using computer vision and to translate arrows seen into lateral directions;
- To accurately land the drone safely on or as close to a final goal position as possible.

1.3 Objective 1

To go into these points in more detail, the first is due to resources. The research does not have the funding or time to build a new drone from the ground up and the idea is to allow current commercial drones to use this technology to prevent incidents. The project will use a Parrot AR 2.0 drone and use JavaScript to program an interface with control for the drone as well as using its on-board cameras to run the computer vision software. The Parrot was chosen over its market competitors for many reasons, the first of which being value for money. For only £170 the drone comes with a 3-axis gyroscope, accelerometer, pressure sensor, magnometer, and ultrasound sensors. The drone also comes with two onboard cameras, one on front, one on the underside. The camera on the front is a 720p, 30fps camera, the bottom is a 240p, 60fps camera. Due to the nature of the project, the risk assessment had also determined the chance of collision in early test stages was at high risk, therefore a drone that would be cheap and easy to repair was also in the criteria. The Parrot is designed to be repaired by the user with all parts sold as spares as well as being sold as a complete drone. The drone

connects via WIFI, creating its own WIFI network on start up using a build in Linux based microcomputer. This is because Parrot have been quoted to want their customers to use them to make mobile game apps, such was said by Felix Geisendörfer at LXJS in 2012. An interface is needed as the first objective, as without it the drone will not fly autonomously, and the system would be a computer vision system for the users viewing.

1.4 Objective 2

The second objective is a simplified version of the original idea. The first test would be to present two arrows in front of the drone, one for up, the other for down, and whichever is closer to the centre of the screen would discern if the drone would need to rise to avoid the collision or drop. This will be further expanded by use of the camera on the bottom of the drone, which will also read arrows on the ground, representing ground objects to move away from or to suggest the drone's optimal route back to its user when a path planning solution is found in future.

1.5 Objective 3

The third and final objective is there for two reasons. The first being that a drone could be above its user when called upon to avoid a risk, therefore a safe landing procedure could be the ideal circumstance. The second being that the battery life of a drone is short, meaning a drone might not have enough power to perform the ideal manoeuvre. In this circumstance it would be required to land a fast, safe landing manoeuvre, preferably as close to its owner as possible. The goal plate would represent this.

1.6 Research Methodology

The project followed the Spiral model for methodology. The Spiral model is a risk analysis-based methodology for project development. In this, the project will plan a prototype, analyze the risks involved with the production, engineer what has been planned, then evaluate the project to find its strengths and how to make it better. Once the evaluation is complete the process begins again until eventually the engineered prototype is fit for purpose. This is ideal for this project as it is one of the safest methodologies in terms of risk prevention, potentially lowering cost and time spent on the project. This is also beneficial towards projects as it highlights the critical goals of a project, as on each rotation of the spiral one can add an additional critical objective on the planning stage.

The Spiral method also has its disadvantages. For instance, if the risk analysis is not completed properly there could be risks that are not calculated, increasing time spent on the project and essentially wasting the time spent on the prior risk analysis. In industry this model can be costly as it can involve many repetitions and risk analysis on each, using a lot of time and as a result money. That being said, there is no financial cost for the manpower on this project, as there is only one person attempting the work.

1.7 Organisation

Chapter two will discuss the literature surrounding this subject and comparing different methods of accomplishing the same goal, and the advantages and disadvantages of such methods. This will cover literature on the control of drones as well as image processing methods. It will then go into detail on other research into precise movement methods and computer guidance methods and understand where these methods have crossovers and providing a critical analysis of these other methods.

Chapter three will cover my own input into computer vision guidance. The algorithm used for the software, the canny algorithm, and the implementation of it to identify arrows and how those arrows translate into commands shall be explained in this section.

Chapter four will cover the precise movement of the drone. This includes calculating movement speeds and how they translate to computational commands, how precise movements have been implemented and the commands to perform these movements.

Chapter five covers the integration of the previous two chapters, how they work alongside each other and the final workings of the overall project. This section will show how an arrow fully translates to a movement.

Chapter six will conclude this project. The final program will be critically analysed and compared to the alternate solutions provided during Chapter two of this thesis. The future expansions for this project and the work required to bring it to its full potential talked about earlier in this introduction will be explained to conclude this chapter.

Chapter Two: Literature review

2.1 Drones

Drone is a commonly used term to describe an unmanned aerial vehicle. These can range in size between the reaper drone, which is the same size as many modern military jets such as the F22 Raptor, to a miniature quadcopter that fits within the palm of one's hand. Drones are also built in various configurations. Some can be planes with a single forward/ rear facing propeller, however the most commercially used version is the quadcopter. The most common configuration has been drawn below:

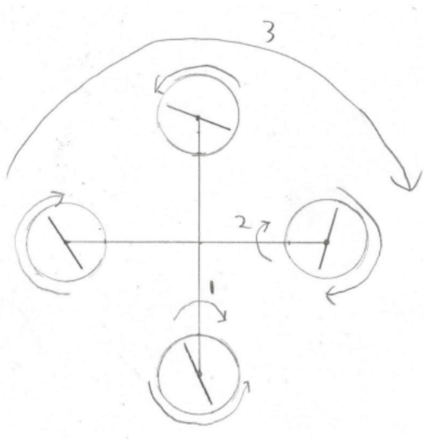


Figure 1, Alex Pollard (2017)

Basic drawing of how a typical quadcopter operates.

This version uses four propellers, with opposite propellers rotating in the same direction and perpendicular propellers rotating in opposite directions. By increasing and decreasing power to specific rotors one can move the drone. To rotate around arrow one on the above image, the left rotor would increase power and the right decrease. To perform the second arrow's movement, the bottom rotor would increase power and the top decrease. The third can be achieved by increasing power to the left and right rotors and decreasing to the top and bottom.

The control of drones using these methods also have various configurations, usually based on the configuration of the drone. The most common such is with a two-stick transmitter using a set radio frequency to connect to the drone. In more recent years it has also become common to use WIFI or Bluetooth to connect to drones for ease of use with smartphones. The Parrot AR 2.0 is one such example of use of WIFI and two-stick flight, although the sticks are simulated on a smartphone. Another version, more common in remote piloting plane style drones such as the reaper, is the use of a side-

stick or joystick. In use of a joystick, one hand controls the yaw and pitch, the other the thrust.

2.2 Image processing

Computer vision is another key factor within the project, in use commonly in many sectors. One of the most seen versions of this is the software powering average speed test cameras. These cameras use a form of computer vision that makes vehicles their focus, using a clever algorithm for real time detection. In this, the cameras would have a database of images to know what a car looks like and is constantly searching for instances of those cars to appear in the image. When one appears, it is highlighted for use.

Another form of computer vision is the use of "Continuously Adaptive Mean-Shift tracking" as described by Shreyamsh Kamate and Nuri Yilmazer. This form of Computer vision focuses on object detection and tracking. This is performed by first identifying the subject and removing the background pixels. This is performed in three stages, the first being background initialisation. Background initialisation monitors the pixels that are remaining constant, or stable. This is called temporal frame differencing, which is calculated as

$$FD_t(x, y) = | I_t(x, y) - I_{t-1}(x, y)$$

(i), Kamate and Yilmazer (2015). Temporal frame differencing.

"$\mathrm{I_t(x, y)}$ is the intensity of pixel (x, y) in the frame at time t "(Kamate and Yilmazer, 2015). Identifying the foreground is found by comparing the frame differencing to a threshold T1. If a pixel's difference is greater than the threshold it is a foreground pixel.

$$FG_t(x, y) = \begin{cases} 1 & if\ FD_t(x, y) > T1 \\ 0 & else \end{cases}$$

(ii), Kamate and Yilmazer (2015) Foreground calculation.

$FG_t(x, y)$ is the foreground pixel. The background pixels must also be stable, so therefore no movement identified. Whenever the frame difference exceeds the threshold, a frame count CF_r is increased by 1, "at which point $CF_r > TF_r$ is analytically determined" (Kamate and Yilmazer, 2015).

$$BM_t(x, y) = \begin{cases} I_t(x, y)\ if\ CF_r > TF_r \\ 0 & else \end{cases}$$

(iii), Kamate and Yilmazer (2015) Pixel stability calculation.

Upon gathering the background data and determining the foreground mask the next stage is to perform background maintenance. To do this, the background difference frame is calculated as

$$D_1(x, y) = |I_1(x, y) - BM_{t-1}(x, y)|$$

(iv), Kamate and Yilmazer (2015) Background maintenance.

The final major stage is to classify the foreground and background. It does this using the background model and current frame. When there is significant motion, these instances are used as the threshold $Th_{Ad,t}$. This threshold and the difference image $D_t(x, y)$ are resolved for each individual frame. Using these a foreground binary mask is generated.

$$FG_t(x.y) = \begin{cases} 1 \; if \; D_t(x, y) \geq Th_{Ad}, t. \\ 0 \qquad\qquad\qquad else \end{cases}$$

(v), Kamate and Yilmazer (2015) Foreground binary mask.

This method of identifying the background has its benefits and flaws. For instance, this method has multiple calculations for each frame for identifying both the foreground and background. This means that if the background is constantly changing, such as it would on any mobile robot, it would require more computational power to identify the background at all times, meanwhile if it were stationary, the background would remain the same requiring computational power to be used on the foreground only. This would therefore be ideally used on stationary cameras. This is likely to be why it is used for tracking unmanned aerial vehicles, as the background in the proposed usage of this would be the sky, which, with exception of clouds, does not change very much meaning less computational power.

2.3 Current Guidance Techniques

Robot guidance has been achieved by many people, using multiple methods. Some methods involve using sensors such as an ultrasonic sensor array on top of a drone, some use forms of computer vision to detect space in front of a drone that is safe to fly into. One such example is that of the work from Wander M. Martins, Alexandre C. B. Ramos, Rafael G. Braga, Luciano do V. Ribeiro and Felix More-Camino. In their journal "Computer vison in remotely piloted aircraft to avoid obstacles during flight" they have simulated a drone collision avoidance method using computer vision. Their method is broken down into seven major stages:

1. Capturing images from the front camera frame by frame
2. Apply greyscale to image
3. Thresholding

4. Divide photo vertically into 3 equal size segments
5. Count the number of black pixels in each segment
6. Move UAV into the area with least number of black pixels
7. Delete old image and repeat with updated image to repeat process.

Their method proposes images being taken by the drone and sent to a computer which is replaced after the photo has been used for movement, similar to the proposed method within this thesis, however their version converts the image to greyscale and analyses the pixels at this point to determine the brightness of areas. It then finds a large enough bright area to move the drone into. This method essentially states that objects are not as bright as open space therefore the brighter the pixel, the safer the flying space ought to be. In practice this method could work as generally objects obscuring the drones path would block out light.

This however has its drawbacks which have not been referred to within their paper. For instance, not all objects blocking the drones path are guaranteed to provide black pixels, for instance this drone would fly into a window or a light if placed in front of it. If this drone were flown at night outdoors this would also have a negative effect as natural light would be limited so the drone would not be able to identify any space as all pixels would come across as black. Alternatively, the drone could be flown in a well-lit street, however the drone would see a street light as being the brightest area and either fly in circles around that or fly directly towards the light itself.

2.4 Precise Movement

When it comes to keeping drone movements as accurate as possible, the number of flight assistive technologies generally increases. In order to perform precision, the ability to keep as still or as slow as possible is crucial. For this reason, many drones with hover functionalities include the use of gyroscopes for monitoring the tilt in all directions. It is also common for these drones to include ultrasonic sensors and pressure sensors, which are needed to maintain altitude. Xiojaun Wei's drone included these and more in the project titled "Autonomous control system for the quadcopter unmanned aerial vehicle". In this document, Wei proposes algorithms for monitoring the posture of the drone and uses this to assist in the control algorithm. The project has many similarities to that of the Parrot AR 2.0, in regard to the near identical technology within his drone, with exception of connecting via Bluetooth rather than the WIFI system within the Parrot.

The control of Wei's drone is governed by the readings of the 3-axis accelerometer and 3-axis gyroscope. When hovering, the gyroscope reads at 0 on the Z axis, the accelerometer reads as 1G to provide and equal reaction to Earth's gravitational pull, and the X and Y axis read as 0. The method involves passing the accelerator value

through a low pass filter and transformed into a three-dimensional unit direction. Using the following two equations, the cosine of the x, y, z axis and reference coordinate system z axis can be converted, which gives the gravitational acceleration of the drone.

Wei acknowledges that the data gathered may have errors, which is why the gyroscope is used for errors elimination. After the experimentation of these code, Wei concludes that the experiment structure is “reasonable”, explaining that the low-level performance was as designed, providing “better safety and portability”. This research is relevant as it works in a very similar way to that of the Parrot drone, therefore this research contributes a better understanding into the workings of the control algorithms for sustained hovering and controlled flight. The why aspect however is lacking. The thesis was submitted in 2016, however the Parrot 2.0 was released in 2012 and comes with all of the same technology as standard, with upgrades available and programming it to perform additional functions is actively encouraged.

2.5 Computer Guidance for Control

Another Method is by assisting the vision-based systems with an array of sensors. For instance, many drones by Parrot include 3 axis gyroscopes, pressure sensors and ultrasonic sensors. These allow Their drones to monitor their stability, altitude and in some cases, their relative position to their take-off position. Ultrasonic sensors are also being used for a project by Bilal Hazim Younus Alsalam, Kye Morton, Duncan Campbell, Felipe Gonzalez. In their project titled “Autonomous UAV with Vision Based On-board Decision Making for Remote Sensing and Precision Agriculture” they have built their drone using Arduino, involving a pixhawk (Pixhawk, 2017) autopilot and ultrasonic sensors to assist in their autonomous flight. Their drone is designed to monitor crop health autonomously, using an originally planned trajectory, however it may change its planned trajectory to perform close inspections of the crops or even apply herbicide. Their system uses an “OODA” loop, or “Observation, Orientation, Decision, Action”. This means the drone is constantly gathering its positional data and comparing it to its proposed data given by its predetermined flight path. This is demonstrated in their paper:

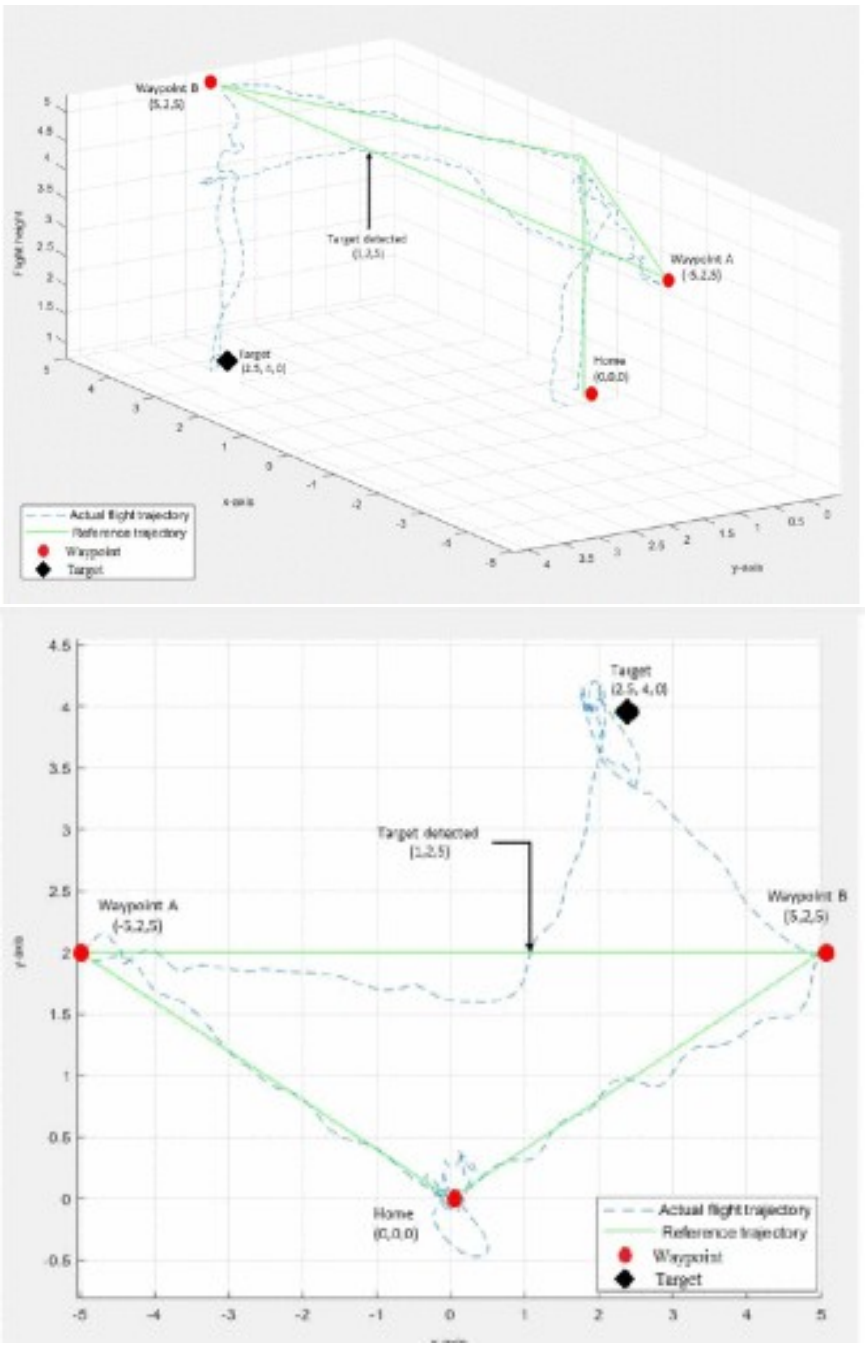


Figure 2, Alsalam, Morton, Campbell, Gonzalez (2017)

Graph demonstrating the movement pattern of the drone

This shows their final experiment with two waypoints, a home position and the target weed for the drone to apply herbicide to. As seen from the image, the drone successfully finds its target after following its predetermined path. The drone has some deviation from its path in areas away from the target, almost always in the same direction. This is likely due to the 15km mentioned in their experiments alongside the

light conditions they suggest weren't favourable. This can also be seen as it took a while for the drone to reach the optimal distance from the target to deploy the herbicide.

From this it can be concluded that the drone's software behind identifying the target, calculating a path to it and control of its autonomous flight is fit for purpose, however it could be fine-tuned with a better system for adapting to the wind. This drone configuration and software therefore is better adapted for outdoor flight than the previously mentioned drone by Martins, Ramos, Braga, Ribeiro and More-Camino. For instance, the Agriculture drone has had real world testing and performed its usage to very accurate levels, while the purely vision-based drone only operated in simulation.

The agriculture drone's technology could also be applicable to the currently proposed topic for flying drones out of high risk areas such as airports due to its ability to identify a plant from its current flight path and navigate towards it. This however would also have drawbacks, as the proposed drone has been built from scratch, which likely means it would be costly to implement into all drones. This is due to the drone being built to fulfil a specific purpose, as opposed to many drones sold which are sold as toys. By implementing their research into commercial drones, it could raise the overall price for their sale.

Another instance of a drone being programmed to use computer vision to assist in guidance is the research by Kevin J. Wu, Thomas Stan Gregory, Julian Moore, Bryan Hooper, Dexter Lewis and Zion Tsz Ho Tse in their thesis titled: "Development of an indoor guidance system of unmanned aerial vehicles with power industry applications". For their project, they fitted four ultrasonic sensors to the top of a Phantom Vision 2+ drone, facing the front, back, left and right. These sensors were used to assist in localisation and real-time navigation. This however presents the issue that there aren't any sensors pointing upwards or downwards, meaning the drone would only be able to detect objects in these four directions. The sensors are set in a continuous loop after initialisation of: Acquire data, yaw correction, package data then transmit data. This information is passed on to the pilot of the drone to assist in their flight.

The drone was tested by six people who were recruited only and did not have a hand in the project itself. This is likely to show their results are not bias by experienced users. Each of these pilots flew the drone four times, with and without the guidance system, and with and without line of sight to the drone. Their aim was to fly the drone from a starting position to a target location. The following data was gathered for each flight:

- Flight duration
- Number of collisions
- Distance from target

Following the testing procedure, they found that with the guidance system active and without line of sight, the flight duration was on average 19.7% shorter, showing that showing a competency of the guidance system. Their results also show that there were no collisions while the guidance system was active at any point, regardless of line of sight. This shows the effectiveness of the sensors. However, the test was conducted based on a two-dimensional plane, and such did not have any overhead obstacles to avoid, showing the potential for further development for the project. Should the drone be able to detect obstacles above and below itself the drone would have potential for use in search and rescue operations in incidents such as collapsed buildings where it would be dangerous for humans to enter.

As many of the above researches indicate, there are many applications for the use of drones in the real world that the average person may not have even considered and the potential for drones is still to be fully determined. The above researches also demonstrate that the technology to develop drones that may identify incoming threats and manoeuvre out of its way is a realistic goal. The research of Wu, Gregory, Moore, Hooper, Lewis and Tse demonstrate this as their drones were not crashed at all, even without line of sight to the drone. Should this sensor technology be applied to the work of Alsalam, Morton, Campbell and Gonzalez, a drone would also be able to plan a route away from the threat autonomously and then, once the threat has been avoided, return back to its pilot, thus completing the project proposed in Chapter one.

Chapter Three: Edge Detection for Computer Vision Guidance

3.1 Canny Algorithm

The Canny algorithm was developed by John F. Canny in 1986. It is a multi-stage algorithm that is used for detecting edges in photos and videos. The first stage is to apply the Gaussian filter which smooths the subject image and removes noise. The gaussian filter kernel size of $(2k+1)x(2k+1)$ has the equation (vii):

$$H_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right); 1 \leq i,j \leq 2k+1$$

(vii), John F. Canny (1986) Gaussian filter kernel size.

After the application of the Gaussian filter, the next stage is to find the intensity gradients of the image. This finds horizontal, vertical or diagonal edges within the image and returns the first value for the horizontal and vertical directions, determining the edge, as in (viii):

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

(viii), John F. Canny (1986) Intensity gradient.

The edge will have one of four directions, vertical at 0 degrees, diagonal at 45 degrees, horizontal at 90 degrees or the other diagonal at 135 degrees. At this stage the third stage, non-maximum suppression.

Non-maximum suppression is the name given to the stage where the thinning of edges is performed. In this stage each gradient pixel is compared to that of the adjacent pixels in either direction. If the pixels in this edge have a greater value than the other pixels in the same direction, the line is kept. If the line is given as a weaker value that the neighbouring pixels, the line is ignored and the values are suppressed. This leaves the image with a more accurate interpretation of the focus of the image. There is still some noise caused by colour variation in the background. To deal with this the fourth stage, Double threshold, selects high and low threshold values. If an edges gradient is between the two threshold values it is deemed a “strong” edge and is kept. All values that fall outside the two thresholds are therefore not kept and are removed.

The final stage is to use hysteresis for edge tracking. What this stage does is examines all of the weak edges previously removes and examines their 8 neighbouring pixels for strong edge pixels. As long as there is at least one strong edge pixel surrounding the pixel, it is kept, all other pixels are removed. The result is an image with a black background with all strong edges represented with white pixels.

3.2 Implementation

The Canny algorithm has been implemented into a JavaScript program. In this program, an image with a predefined name is uploaded onto the GUI, presented in HTML form. Upon receiving the image, the software converts it into greyscale in the form of a matrix.

Once the picture has been converted into a matrix, the pixels are then constructed into a new image in greyscale:

```
function GrayImageData(width, height) {
    this.width = width;
    this.height = height;
    this.data = Util.generateMatrix(this.width, this.height, 0);
    this;
  }
```

The greyscale image is then presented in a canvas in html, this method was chosen instead of a GUI for simplicity. The original image would be placed in another canvas simultaneously. This would ensure that the drone's performance could be monitored if needed.

```
GrayImageData.prototype.drawOn = function(canvas) {
    var color, ctx, i, imgData, _i, _len, _ref;
    ctx = canvas.getContext('2d');
    imgData = ctx.createImageData(canvas.width, canvas.height);
    _ref = this.toImageDataArray();
}
```

Once the image data was given in greyscale, the canny algorithm was ready to implement. While testing the figures for the ideal high and low threshold, sigma and kernel sizes, the use of a file on Github by author "Yuta1986" was necessary. Yuta's program was implemented to run different pictures than the three demonstration pictures, namely, three photos taken by the drone of two arrows and the landing zone image created for the project. Once the preferred settings had been found, the figures were inserted into the projects code to begin testing.

The gaussian blur was then applied, alongside the figures for the sigma, high and low threshold and kernel size, which were 1.4, 100, 50 and 3 respectively. As explained in Chapter 3.1, The gaussian blur is calculated as:

```
for (j = _j = 0, _ref1 = size - 1; 0 <= _ref1 ? _j <= _ref1 : _j >= _ref1; j = 0 <= _ref1 ? ++_j : --_j) {
      y = -(size - 1) / 2 + j;
      gaussian = (1 / (2 * Math.PI * s * s)) * Math.pow(e, -(x * x + y * y) / (2 * s * s));
      kernel[i][j] = gaussian;
      sum += gaussian;
    }
```

```
    return copy.data[x][y] = Math.sqrt(ghs * ghs + gvs * gvs);
```

This shows, although the symbols have been exchanged for letters, the equations are near enough identical. Once the Gaussian filter has been applied the non-maximum suppression is performed. This is where the thinning of edges takes place, so each pixel is compared to the kernel value:

```
  imgData.eachPixel(3, function(x, y, c, n) {
    if (n[1][1] > n[0][1] && n[1][1] > n[2][1]) {
      copy.data[x][y] = n[1][1];
    } else {
      copy.data[x][y] = 0;
}
```

The next stage is the application of the hysteresis, where all pixels are compared to the high and low threshold. All edges that fall within the threshold are traced along to provide the final image:

```
 CannyJS.hysteresis = function(imgData, ht, lt) {
   var copy, isCandidate, isStrong, isWeak, traverseEdge;
}
```

To test this, a random image from the authors camera roll was put through the edge detection system, resulting in the following image:

Figure 3, Alex Pollard (2017) Author

Image of drone that has been processed by the Canny Edge detection.

The image was of the parrot drone sitting on a striped bedsheet, which wasn't the ideal background for identification of an object but had a lot of noise. Although these stripes are clearly visible in the final image, the drone is also clearly visible. This shows that if an arrow were placed on the ground, regardless of the floor, the drone should still be able to see the arrow through any noise.

The following images were created to use as directions for the drone to follow, the images had multiple edges to make it easier for the drone to detect the arrows should they be placed on a similarly coloured floor.

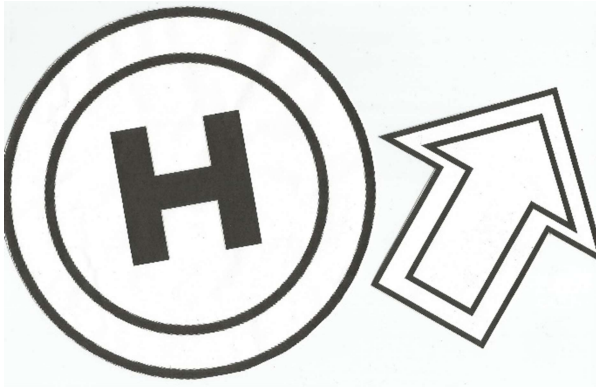


Figure 4, Alex Pollard (2017)

Landing Zone and Direction arrows, printed to be place on floor.

Chapter Four: Precise Movement Implementation and Testing

4.1 Initial Connection

The Parrot drone is connected via its own created WIFI network to a laptop which, for testing purposes, will run the image recognition described in Chapter 3.3. Now the drone is connected, a client to run the drone is created. This client will receive commands from the image recognition software and convert these into movements. Before any of this can be accomplished, the movements themselves must be defined and all background information needed to fly must also be prepared. The client tells the drone to send all navigation data collected to the laptop, which is presented in the command prompt continuously. This navigation data is important as it shows us where the drone thinks it is in comparison to its take-off position, which if done correctly, will be identical to its real-world position. The final version of the code is written in JavaScript. This is due to the native language of the drone which is also in JavaScript, therefore assisting with compiling the code.

The drone is then given its different states of motion, which are landing, landed, take off, hovering and flying. While in take-off and landing states, the drone will not acknowledge any information given to it until these acts are complete. This is due to safety reasons, as it would not be safe for the drone to attempt complex manoeuvres until it is in a stable flying position. This is similar to when it is in the landed state, as in this position it will not respond to any movement-based commands other than the take-off command. The drone therefore only runs the movement commands while in the hovering or flying states, however to maintain precision the drone will receive commands after each movement to stop and search for the next target, therefore the majority of the commands should be received in the hover state only, though it is not impossible for it to receive a string of commands while still in flying mode.

4.2 Movement commands

The Parrot drone responds to the following movement commands:

- Takeoff, Land
- Front, Back
- Left, Right
- Up, Down
- Clockwise, CounterClockwise
- Stop
- Emergency

The syntax for the above commands follow a simple structure:

```
client.after(6000, function() {
        this.front(0.4);
        });
```

The above commands tell the client running the drone that the drone should wait 6 seconds (6000 milliseconds) then move forward with a speed value of 0.4. This speed value translated to four hundred millimetres per second once the drone had fully accelerated. This was a good speed to work with as the drone wouldn't take tong to accelerate or decelerate at this speed, minimising the inaccuracy in movement.

The take-off command initiates the drone, changes its state from landed to take-off, then starting all four motors and brings the drone up to one metre altitude. Upon reaching one meter the drone enters the hover state. In the hover state the drone is ready to receive movement prompts. This is effectively the opposite of the land prompt, which takes the drone from flying or hovering states and puts it into a landing state. While in this state, the drone lowers the propellers rpm to provide a controlled descent. Once the sensors on the underside read that it has made contact with the ground, the propellers come to a complete stop and the drone comes to a landed state.

The nose mounted camera marks the front of the drone, so all directions given to the drone are made from its perspective. In order to move forward the rear propellers are accelerated, and such happens for each other lateral movement, where the propellers on the opposite side of the drone are accelerated to reach the desired movement. While moving in any direction the drone automatically changes its state from hovering to flying. A drone can be returned to the hovering state by giving the stop command. In order to rotate on the spot, the clockwise and counterClockwise commands are given. These commands make opposite propellers rotate faster than the perpendicular propellers. By doing this, the air current provided by the two faster spinning propellers increases, providing a rotary motion of the whole drone. This works as opposite propellers spin in the opposite direction to their perpendiculars, and as such, are shaped as a mirror image too.

The drone also comes with two built in safety features, which can be controlled to a degree. The first is the emergency mode, which kills power to all motors, forcing a crash landing. This is activated when the drone has either already crashed or has tilted ninety degrees in any direction. This is to prevent the drone from accelerating away in extreme winds. The reason it kills power is due to self-preservation. The developers at Parrot knew that if a drone crashes with its propellers running, the propellers

continually hitting the surface would cause more damage than if the drone were inactive. The drone will usually land on either its propellers which are cheap, or the landing gear or outer casing, all of which are cheap to replace, however by landing on the propellers with them still active will likely damage the motors too, and due to increased damage, the option of there being plastic shrapnel from the broken propellers is more likely.

The other built in safety feature is the low power mode feature. The drone monitors its own battery, even when it is not being displayed within a program. Once the drone reaches thirty percent maximum power the drone will not allow its user to attempt to perform a flip. If the drone is landed and drops below thirty percent, the drone will also prevent the user from taking off, as that is the most power consumptive stage of the flight procedure.

4.3 Precision

The movements above may be given in two forms. The entire flight can be performed based on a timer, or the instructions can be given on command, and will be executed continuously until given a new command. This is important as it was critical for creating the precise movements needed for the final iteration. When flown with a timer, it takes five seconds, or five thousand micro seconds as the drone reads the time, to take off and land. This is relevant as knowing time means the use of the SUVAT equations may be used to determine acceleration, which is an important factor to consider when achieving precision.

Knowing only acceleration is not enough however, as in order to calculate the other figures there must be at least three known variables. An experiment was planned to plot the other figures. Two gates were placed nine feet (2.7432m) apart, precisely one and a half metres wide and the drone was set to hover directly at the first gate. The drone was timed based on when the drone passed through the gates according to its own camera, therefore reducing human error for reaction times and point of view errors that may occur based on where one stands. The test was conducted indoors with no other people other than the pilot in the room. This was repeated ten times.

Attempt number	Time (s)
1	1.50
2	1.30
3	1.44
4	1.48
5	1.46
6	1.47
7	1.44
8	1.49
9	1.45
10	1.44

Table 1, Alex Pollard (2017)

Table for testing for precise movement calculation

The average time was therefore 1.447. The second figure appears to be an anomaly, being .14 of a second faster than any other instance. This is likely due to a pre-emptive tap of the stopwatch rather than the drone picking up speed for a single flight, considering how the other results all fell within 0.06 of a second from lowest to highest. The average time is then taken and applied to the equations of motion. The equations of motion are as follows:

$$V = U + AT$$

$$S = UT + \frac{1}{2}AT^2$$

$$S = \frac{1}{2}(U + V)T$$

$$V^2 = U^2 + 2AS$$

$$S = VT - \frac{1}{2}AT^2$$

(viii), Equations of Motion (SUVAT)

The Equations of motion are based on 5 factors and using a combination of 3 known factors it is possible to work out the remaining two. The factors are represented by five letters; S, U, V, A, and T which represent distance, initial velocity, final velocity, acceleration and time. The test was designed so that 2 factors were guaranteed to have been known and third being measured, these were distance and initial velocity, with the time being measured. The acceleration can thus be calculated using the second equation listed above:

$$2.7432m = 0 + \frac{1}{2} A\ x\ T^2$$

$$5.4864m = A\ x\ 1.447$$

$$A = 3.792m/s^2$$

Now that the acceleration is known, any amount of distance can be provided using the same power constant and a time given to a thousandth of a second. The figure has been rounded to four places to keep as much accuracy as possible, as the system operates time in thousandths of a second. The maximum speed on the drone is able to be set, allowing further control of the drone. As this work requires precision, and indoors the drone has minimal drift, the slower the movement the better. The drones top speed was set as three metres per second.

To generate a one metre movement, the following equation was used:

$$1 = 0 + (\frac{1}{2} x\ 3.792)\ x\ T^2$$

$$1 = 1.896\ x\ T^2$$

$$0.527 = T^2$$

$$T = 0.726$$

To confirm this, the drone was programmed to accelerate forward for precisely 0.726 seconds then come to a complete stop. Using a tape measure two positions were marked precisely one metre apart, with a person on either side to check the exact final location and exact start location. Once the drone was in the exact start location, the program was initiated.

The test provided information of another factor that hadn't been considered, that the drone doesn't start accelerating until it has fully tilted. Although it was true that the drone accelerates at $3.792ms^2$, the drone also needed time input to lean into the acceleration, which was discovered after running an additional four runs, brought the acceleration time up to 0.9 of a second.

Chapter Five: Integration

5.1 Final Version Design

Once the basic functions were up and running, the next stage was to implement them together to produce the final version of the software. To do this, a flow chart was created to help with the functionality design of the software.

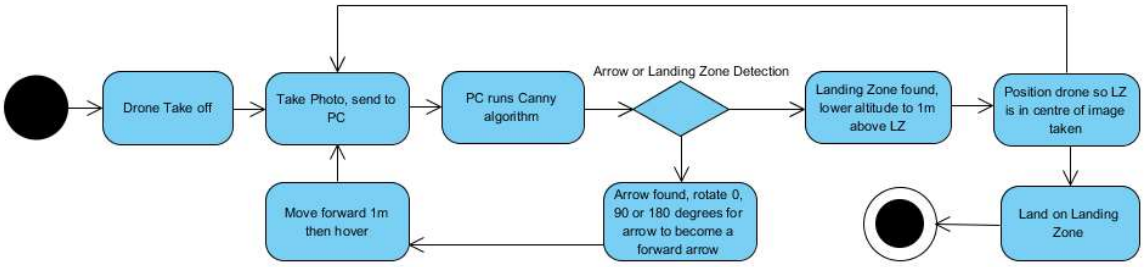


Figure 5, Alex Pollard (2017),

Flow chart showing design of system function.

Upon initialisation of the software, the drone would take off and take a photo. This photo would always save with the same name, which overwrites the previous versions of the image to make sure the current picture is always being read. Upon taking the photo a web browser opens with two images side by side. The one on the left being the image just taken by the drone, and on the right is the same image which has had the Canny Edge Detection applied. In the background the software compares the Canny image to the database, which only contains the known images with directional prompts. If one of the directional prompts matches the image seen the system will move in the direction recognised or land if a landing point is the image seen.

5.2 Initial Testing of Final Version

For the initial testing of the project all arrows and landing zones will be placed precisely one metre apart. The path would involve one of each direction to demonstrate the drone's ability to apply the changes in direction. This is also due to a minor note during testing that the drones directional control was best while moving forwards. During the speed testing it was noted that over the nine feet of forward momentum the drone passed through the centre of the first gate and passed roughly five centimetres to the right of the centre on the centre gate, although the drone did face directly forward. This is likely due to a small section of damage to the forward right propeller protector, resulting in an imperfection in the aerodynamics. Ideally, this would be replaced, however the project does not have the budget to replace it.

The drone was set up to stop after each command given to it, so to begin the program, the drone was told to take off, then given no other movement instructions, resulting in a hover. The drone would then take a photo, which would always be saved in the same folder with the same name, overwriting the previous photo to avoid a repeated instruction. This section of the code was run through a node.js command prompt. Here is the first image taken by the drone on the initial test:

Figure 6, Alex Pollard (2017),

Arrow photo taken by drone

The canny Algorithm was housed in a separate code which ran its JavaScript on a html page. This page displayed both the photo taken by the drone and the photo after the canny algorithm was applied to the photo. The initial test saved the first photo (Figure 16) successfully, and the canvas with the canny implementation displayed Figure 17 (below).

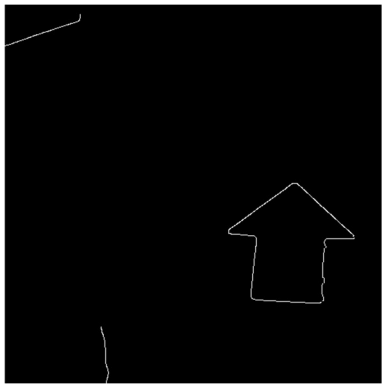


Figure 7, Alex Pollard (2017),

Figure 6 arrow photo after Canny Ede Detection has been applied.

As seen, the arrow that was placed on the floor is clearly identifiable to the human eye. This shows the initial test was a success. The drone was then instructed to land via commands sent to the drone manually to prepare for the next test. It was realised that although the precision flight had been successfully programmed, and the canny

algorithm had been implemented correctly, no database had been created for the algorithm to compare it to.

5.3 Conclusions

To reflect upon the original aims and objectives:

- To implement computer control for drones and use this to run the program through the drone;
- To get the drone to recognize directions given by arrows using computer vision and to translate arrows seen into lateral directions;
- To accurately land the drone safely on or as close to a final goal position as possible.

Chapter four: Precise Movement demonstrates that the drone has been successfully programmed to fly using prompts from a program, using node.js. The execution of this was smooth and produced almost no issues while testing, the only exception being down to a human error. A note of all controls was made during the initial testing as to make it easier to reference them during final iterations of the project. Once the movements had been calculated, times, distances and speeds were made note of in the bottom of the program, as to make it easier to reference to them later. With this in mind, it can safely be said that the first objective has been fully accomplished.

The test for providing a metre of movement provided the answer for the third objective. When testing the metre movement described also in Chapter four, it was discovered that not only did the drone move precisely a metre forward, but when instructed to land, it stayed perfectly straight when performing its landing procedure. With that information, it is safe to assume that had the database been implemented correctly the third objective would have been completed in its entirety. Without the database referencing however, the drone would have no way of accurately checking this. It can be said however that the drone would land precisely on the pad if the flight path has been programmed into the drone prior to taking off.

The second objective however was not a complete success. The drone was able to send the photos and the computer was able to remove the majority of noise from a photo to present a clear arrow for direction, however the translation of that image to a direction had been a failure. With that said, the actual prompts to send the drone did work however. This was tested by inserting buttons onto the html page with the direction commands. Once pressed the drone would move a metre in that direction and then stop to take another photo, which was then updated. This shows that although the second objective was not completed, it had the potential for completion.

If the project were repeated knowing what is known now, rather than the use of HTML and JavaScript for the implementation of the Canny algorithm, the use of an open

source software such as OpenCV would have provided an advantage for this project, as it is purpose built for running computer vision projects and would have been easier to implement.

Chapter Six: Further Work

6.1 Further work

One such way the project could be progressed towards the final goal would be to implement the work of Youmin Zhang and Abbas Chamseddine. Their project was to consider actuator faults and propose fault tolerant control methods to accommodate these effects on the systems performance. Their research tested 6 strategies, two of which were tested in simulation and real-world experiments without any damage. This research could be relevant as a drone could have been flown off course and into a hazardous area, such as getting too close to an airport, due to damage sustained during flight. With their methods this risk could be prevented or controlled better.

Another key aspect critical to the project is the path planning. Once the drones have their workspace mapped out and their own positions are known they would need to know the various routes and target destinations for their tasks. This is where path planning software would be critical. In Nikolaus Correll's research paper, "Introduction to Autonomous Robots", he covers 4 path planning methods. These are Dijkstra, A*, D* and RRT. After reading through this paper, the ideal algorithm to follow would be D*, as it doesn't waste computing power looking in the wrong direction straight away, unlike Dijkstra, which investigates every possible route and its cost. It also has an advantage over A* as it allows for recalculating around obstacles on the path. This could also present problems as the examples shown are in two-dimensional space, meanwhile the drone will require a full three-dimensional environment. Path planning would be a crucial element in the future development of the project as a drone could see the risk using software currently in use, then activate the path planning software to find the optimal route away from the risk and then back to its owner.

This highlights key aspects that would be critical to the development of drone technology. Were the research to continue, the development of advanced path planning technology would be the logical next step. The drone is already capable of following instructions via only visual aids to find its landing zone, so if the drone used natural images such as a tree to recognize that it was off course this would be a step in the right step towards the ultimate goal.

References

Ben Quinn (11/10/2015), 'UK police see spike in drone incidents, The Guardian' https://www.theguardian.com/technology/2015/oct/11/drone-incidents-reported-to-uk-police-on-the-rise

UKAB (22/02/2017) 'Assessment sheet for UKAB meeting on 22nd February 2017' https://www.airproxboard.org.uk/Reports-and-analysis/Monthly-summaries/2017/Monthly-Meeting-February-2017/

Felix Geisendörfer (29/9/2012) 'LXJS 2012 Programming flying robots with Node.js', https://www.youtube.com/watch?v=jl5v3bsMH_E

Youmin Zhang, Abbas Chamseddine (2012) 'Fault Tolerant Flight Control Techniques with Application to a Quadrotor UAV Testbed'. Department of Mechanical and Industrial Engineering, Concordia University, Canada.

TAN, Y + ZHENG, Z.Y (2013) 'Research Advance in Swarm Robotics'. Defence Technology, 9(1) 18-39

Shreyamsh Kamate, Nuri Yilmazer (2015) 'Application of Object Detection and Tracking Techniques for Unmanned Aerial Vehicles'

Thomas Endres (13/01/2013) 'Windows AR Drone Control Software', http://www.winardrone.com/

Jamie Condliffe (10/01/2017) 'A 100-Drone Swarm, Dropped from Jets, Plans Its Own Moves' https://www.technologyreview.com/s/603337/a-100-drone-swarm-dropped-from-jets-plans-its-own-moves/

Wander M. Martins, Alexandre C. B. Ramos, Rafael G. Braga, Luciano do V. Ribeiro, Felix More-Camino (July 2017) 'Computer Vision in Remotely Piloted Aircraft (RPA) to Avoid Obstacles During Flight', International Conference on Advanced Robotics (ICAR)

Bilal Hazim Younus Alsalam, Kye Morton, Duncan Campbell, Felipe Gonzalez (2017) 'Autonomous UAV with Vision Based On-board Decision Making for Remote Sensing and Precision Agriculture'

Nicholas Correll (2016) 'Introduction to Autonomous Robots', chapter 4, Path Planning.

Xiaojuan Wei (19-22/08/2016) 'Autonomous control system for the quadrotor unmanned aerial vehicle' 2016 13th international conference on ubiquitous robots and ambient intelligence.

Kevin J. Wu, Thomas Stan Gregory, Julian Moore, Bryan Hooper, Dexter Lewis, Zion Tsz Ho Tse (8/5/2016) 'Development of an indoor guidance system of unmanned aerial vehicles with power industry applications'. IET, Institution of Engineering and Technology journals

yuta1984 (6/12/2014) ‘CannyJS’, https://github.com/yuta1984/CannyJS

Lorenz Meier (10/08/2017), ‘Pixhawk’, https://pixhawk.org/